|  |  |
| --- | --- |
| **DevOps**  Diploma in IT  Year 3 | Week **2** |
| **2** hours |
| **Week 2 Exercise** | |

**Part A.1 – Preparation**

For the ease of understanding and accountability, the following shall be used for group work.

* **Repository name:** *DevOps\_Oct24\_Team<number>\_Prac<Number>*
* **Language of choice:** Python (file extension .py)
* **Extra Collaborator to add:** erpv-np

Please note that this is a team shared repository so all members of the group would need to be added as collaborators.

All created repositories are defaulted as public repositories unless specifically stated.

**Git Account Registration**

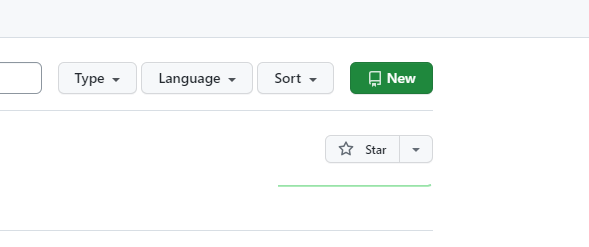
Register for a Git account if you have not done so.

* <https://www.github.com>
* Please note that this is a personalized account that would be used potentially as a profile during employment.
* **Avoid using unnecessary names** that do not officially associate the account to you as a potential interview candidate. For example, MyHimePrecious2020, IForYou2020 or s01261313 etc.
* Refer to <https://www.linkedin.com/learning/craft-a-great-github-profile/create-a-great-github-profile?autoplay=true&u=42538748>

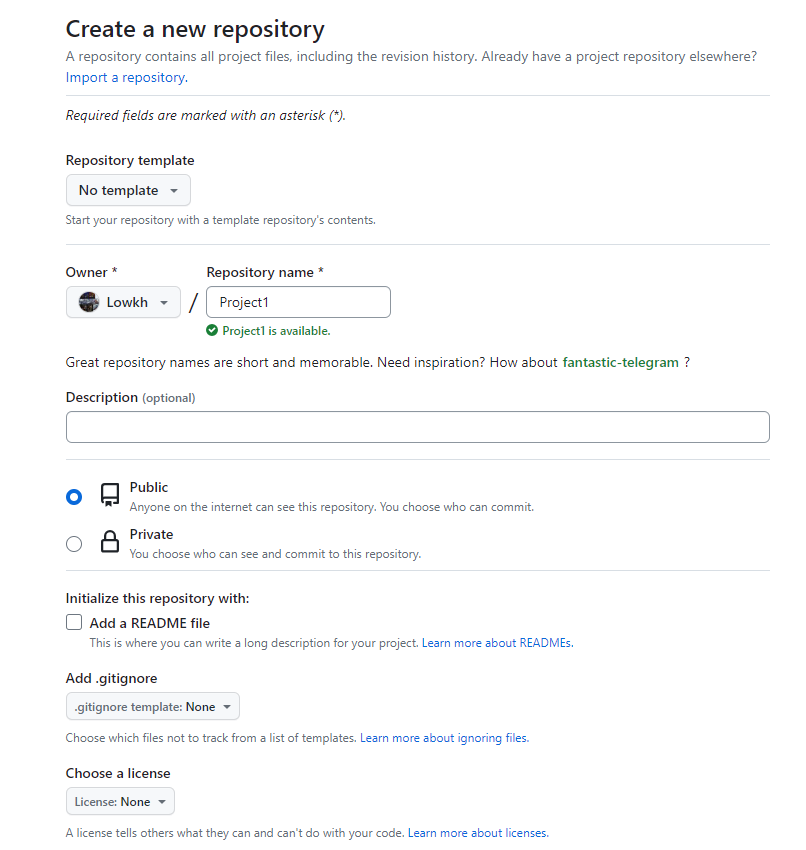
**Part A.2 – Git Repository Basics via Console**

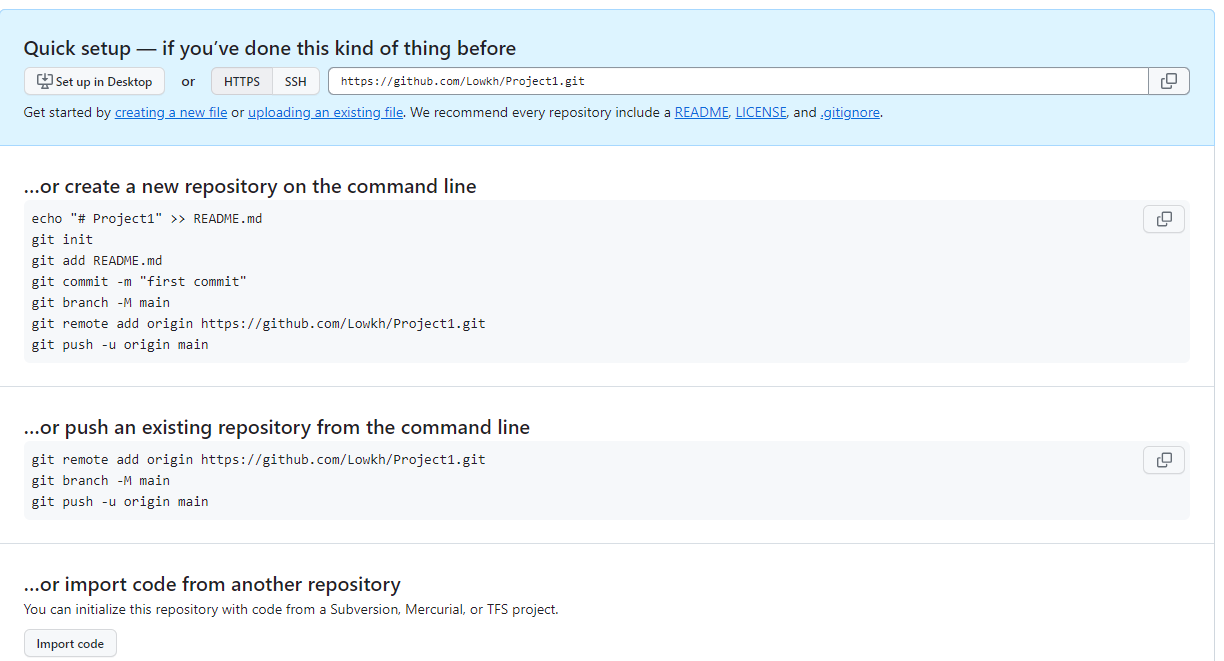
Start your command prompt using

* “CMD” for windows / Terminal for Mac
* Create a working directory for your project on your machine
  + Use command to create a directory - ***mkdir <Project Directory>***
  + Use command to navigate to the directory – ***cd <Project Directory>***
  + Use command to create a subdirectory – ***mkdir <Project 1 Directory>***
  + Use command to navigate to the directory – ***cd <Project 1 Directory>***
  + Suggestion on the directory Documents/DevOps/Exe1
* Initialize Git in the working directory
  + Use command to enable git – ***git init -b main***
* Go to Github and log in to your account. Create a new repository on GitHub by selecting “New”.



* Give the project a name and ensure the following
  + Set to Public.
  + Do not select “Add a README file” (Optional)
  + Select “Create repository”

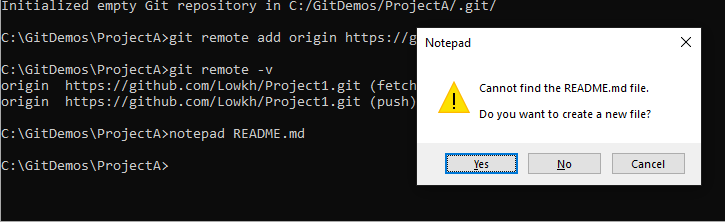


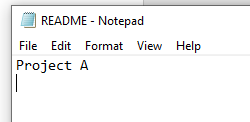


* Associate the local working directory on your machine by using the following command as highlighted in red.
  + ***git remote add origin https://github.com/<your github username>/Project 1.git***
* Issue the following command to verify the association is correct.
  + ***git remote -v***

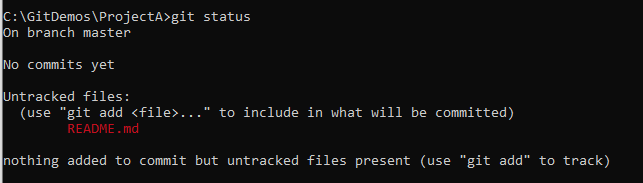


* Create a new README.md using command – ***notepad README.md***
* Create content for it and save.

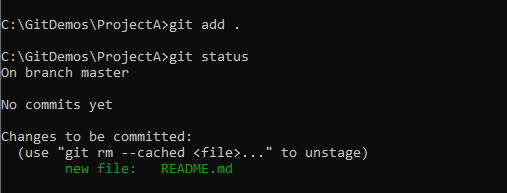




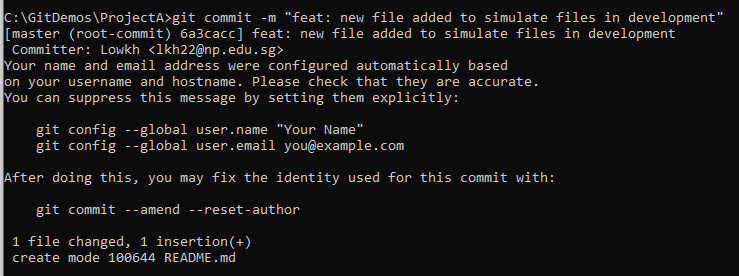
* Use command – ***git status***
* Observe that the new file is currently untracked.



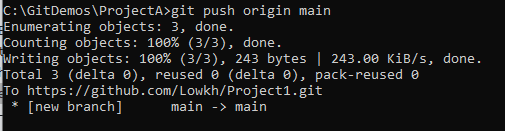
* Use command – ***git add . (Mac OS use command – git add README.md)***
* Use command – ***git status***
* Observe that the new file is currently tracked



* Use command – ***git commit -m “feat: new file added to simulate files in development”***



* Depending on your machine settings, you may or may not see the warning regarding configuration of username and email address.
* Use command – ***git push origin main***
* This would push the new file into the remote repository. Refresh the repository in the browser and observe the changes.



**Part B.1 – Git Repository and Version Control via Console**

Create a new repository and add team members in.

* Create a new repository.
* Create a basic hello\_world.txt and commit it into the team repository with the comment “Initial Commit”.
* Add members into the newly created team repository as collaborators.

**Part B.2 - Collaboration Git Updates**

Collaborate with the team via a simple console activity.

**Part B.2.1 - Basic Pull, Branch, Merge, Push**

For each member

* Clone the team Git repository into your local machine.
  + To clone repository
    - use command – ***git clone <git url>***
* Create a local branch using a meaningful name
  + To create branches
    - use command – ***git branch <branchName>***
    - use command – ***git checkout –b <branchName>*** (create and checkout to the branch)
  + To delete branches
    - use command – ***git branch –d <branchName>***
  + To list out all branches
    - use command – ***git branch***
* Checkout to your corresponding branch and add the following code in the file.
  + *“This is hello from <insert your name here>. I am adding a new line of code here.”*
* Commit your additions with the following message
  + *“feat: basic commit with added new line of string”*
* Modify the code again to add the following line to be printed
  + *“Hello again. I am here to add in a second line. I am <insert your name here>”*
* Commit your additions with the following message
  + *“fix: basic commit with added second line of string”*
* Switch back to the main branch and prepare to merge the branches.
  + To merge from the branch to the main locally
    - use command – ***git merge <branchName>***
* Push your codes back to the repository directly
  + - use command – ***git push***

Observe the different conflicts induced by other team members if any. Attempt to resolve the conflicts by the following actions

* A pull from the repository is needed to extract the latest changes induced by others to the local repository.
* Compare the difference that is flagged out.
  + The code will be separated by “<<<<<” and “>>>>>>” demarking the different versions of code.
  + Select to add on both codes and attempt another commit and push to the repository.

**PART B.2.1.1 - Evidence of work**

The exercise should eventually contain the different names of all the members. Do a git log and submit your screenshots in Brightspace.

All members of the team can also do a fork of the Git for further references if necessary.

**Part B.2.2 Basic Pull, Branch, Push + Merge UI in Git**

For each member

* Clone the team Git repository into your local machine.
  + To clone repository
    - use command – ***git clone <git url>***
* Create a local branch using a meaningful name
  + To create branches
    - use command – ***git branch <branchName>***
    - use command – ***git checkout –b <branchName>*** (create and checkout to the branch)
  + To delete branches
    - use command – ***git branch –d <branchName>***
  + To list out all branches
    - use command – ***git branch***
* Checkout to your corresponding branch and add the following in the code in the file.
  + *“This is hello from <insert your name here>. I am adding a new line of code here for direct branch push.”*
* Commit your additions with the following message
  + *“feat: commit for branch direct push with added line of string”*
* Modify the code again to add the following line to be printed
  + *“Hello again. I am here to add in a second line for direct branch push. I am <insert your name here>”*
* Commit your additions with the following message
  + *“fix: commit for branch direct push with added second line of string”*
* Push your codes back to the repository directly from the branch
  + use command – ***git push –u origin <branchName>***

Observe that the branches are pushed in as branches to the repository. From the Git repository, select Pull Request and merge selecting the main branch and the branch to merge with.

* Compare the difference that is flagged out if any.
  + The code will be separated by “<<<<<<” and “>>>>>>” demarking the different versions of code.
  + Select to add on both codes and attempt another commit and push to the repository under the main branch.
  + Leave comments using Git Pull Request setup and ensure a trail of communications has been established.

**Part C.1 – First Project**

Git Projects can be used to track and prioritize different tasks needed for the project. This is a replacement of the physical AGILE/ Kanban boards that were used traditionally. GitHub provides us with semi-automated boards that makes project management easier in Git.

**Creation of My First Project**

* Create a new repository and add team members in.
* Create a new repository.
* Select the option for “README.md” to be created too.
* Add members into the newly created team repository as collaborators.
* Refer to <https://docs.github.com/en/issues/planning-and-tracking-with-projects/creating-projects/creating-a-project>
* Prepare a simple project and name the project “Simple Calculator”

**Part C.2 – Scrum Online**

We would like to create a simple scientific calculator. We should develop it with Python. Note: This code will be used in subsequent practical. Do spend some time to design and develop the codes.

The calculator should contain the following abilities.

* Ask the user for an input to select the functions needed.
  + Standard Arithmetic.
    - Addition/ Subtraction/ Division/ Multiplication
  + Trigonometry.
    - Sin/ Cos/ Tan/ Inverse Sin/ Inverse Cos/ Inverse Tan
  + Exponents.
    - Power of X
  + Square Root
  + Percentage
* Ask the user for the relevant value input.
* Display the result.
* Display any relevant errors if any. For example, division by zero.

**Part C.3 – Project discussion and tracking setup**

* Refer to the following videos from GitHub:
  + <https://www.youtube.com/watch?v=qT0VMdx7vuI>
  + <https://www.youtube.com/watch?v=yFQ-p6wMS_Y>
* Discuss and decide on the different columns/ labels/ sequence of events that are necessary for the Git Project.
* Establish a flow of events and operating processes for the use of the Git Project within the team.
  + **NOTE: This would likely be what the team shall follow when doing software development. You may change and improve the process as the module progresses.**
* Decide on how an issue is generally translated across from requirements/ user stories to tasks and eventually assigned to the relevant members of the team.
* Update the considerations and processes in the Git readme.
* Consider the existing system of labels used and decide if the team requires more labels to separate tasks.
* Refer to <https://docs.github.com/en/issues/using-labels-and-milestones-to-track-work/managing-labels>

**Part C.4 – Project Allocations**

* Refer to <https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues>
* Refer to <https://www.linkedin.com/learning/scrum-the-basics/practicing-scrum-in-your-work-environment?autoplay=true&resume=false&u=42538748>
* Refer to <https://www.linkedin.com/learning/scrum-advanced/move-your-scrum-team-into-high-gear?autoplay=true&u=42538748>
* Discuss and populate the board with the requirements / user stories of the activity.
* Appoint the scrum master role, each member of the team (including the scrum master) is to proceed and do the assigned task.
* Address each issue with the corresponding tag (i.e., development, enhancement etc.) accordingly to attempt to close all opened issues as far as possible.

**Part C.5 – Stock take**

* Monitor the project columns as the project progresses.
* Take relevant stock take of the project and submit a simple progress report at the end of the lesson. i.e. Burn up/ backlog/ issues closed etc.

**PART C.5.1 - Evidence of work**

The exercise should eventually consist of work done and a trail of development evidence. Do a git log and submit your screenshots in Brightspace.

All members of the team can also do a fork of the Git for further references if necessary.

**Part D – Further information/ references**

For those who are interested to understand how to do a full-fledged Github Project Board with Reviews in place, refer to the following for a quick start.

<https://www.youtube.com/watch?v=YxKhb3fxtsU&t=559s>

<https://medium.com/@gitship/15-git-hacks-to-save-your-life-as-a-developer-aa8808846dbb#:~:text=To%20exit%20git%20log%2C%20type,h%E2%80%9D%20to%20seek%20for%20help>.

-END-